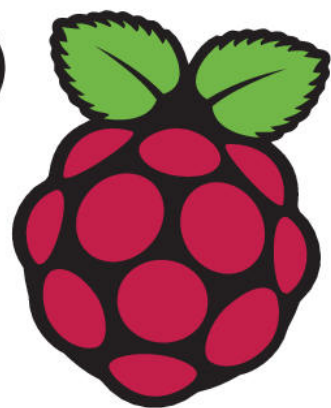
LA RIVISTA UFFICIALE TRADOTTA IN ITALIANO

La rivista ufficiale Raspberry Pi in italiano, da RaspberryItaly.com

Numero 64 Dicembre 2017 www.raspberryitaly.com

ELETTRONICA: GUIDA DI PARTENZA

Vero divertimento con componenti, circuiti, e cavallotti

IN ARRIVO!

AIY PROJECTS VISION KIT

Made by you with Google

REALIZZA UNA AVVENTURA TESTUALE

Gratuito!

VISION KIT

La squadra Google AIY Projects svela la fotocamera intelligente fai-da-te per Pi Zero W

Google AIY Projects ha annunciato il suo Vision Kit, un kit smart camera che puoi realizzare usando un Pi Zero W, un Pi Camera Module, e una potente scheda IA chiamata 'Vision Bonnet'.

I fan appassionati di Google riconosceranno la somiglianza tra il Vision Kit e Google Clips, la fotocamera intelligente recentemente annunciata.

Il Vision Kit consente ai maker di costruire un dispositivo simile, ma questo è solo l'inizio. Tutti i progetti AIY sono sullo sviluppo di kit di IA hackerabili per i maker, che possono integrarli nei loro progetti.

"Manteniamo l'attenzione su come mettere queste tecnologie in mano alle persone in modo che siano facili," dice Jess Holbrook, AIY Projects UX Research Lead. "La IA e l'apprendimento automatico sono su questo piedistallo", spiega. Google vuole

IL VISION BONNET

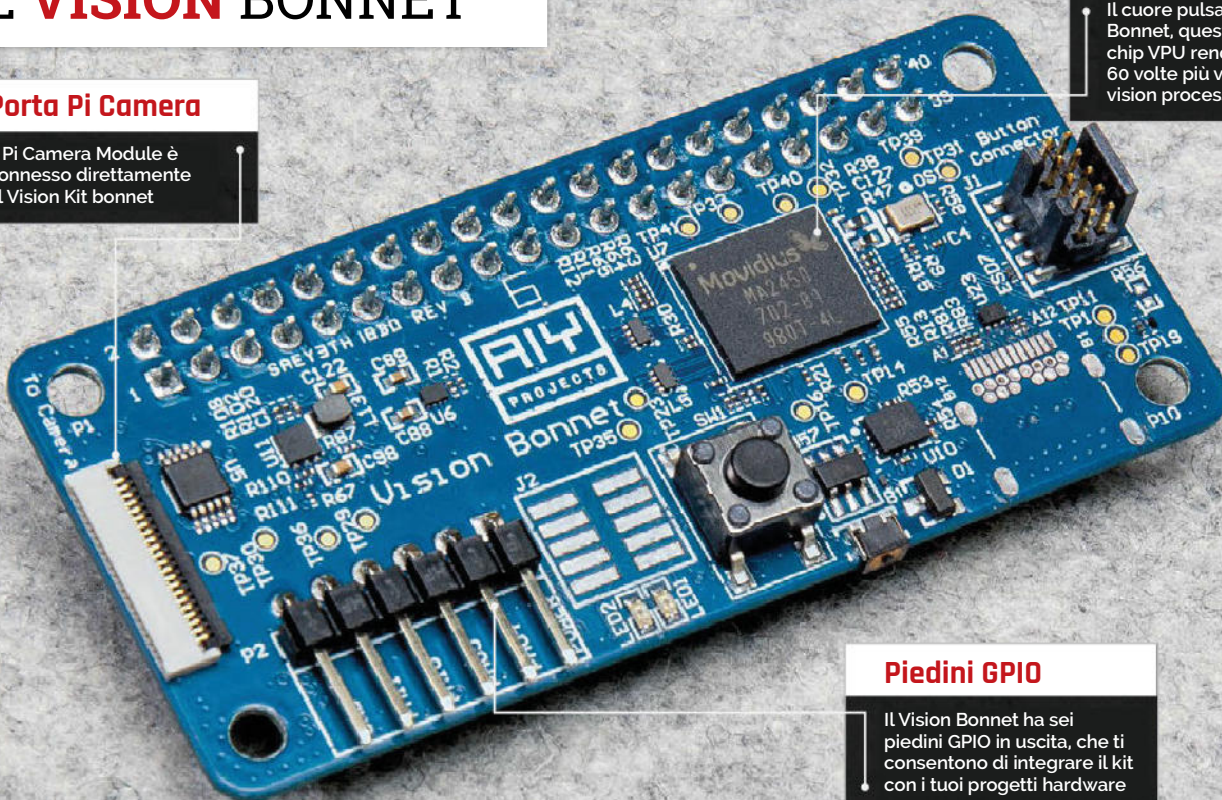
Porta Pi Camera

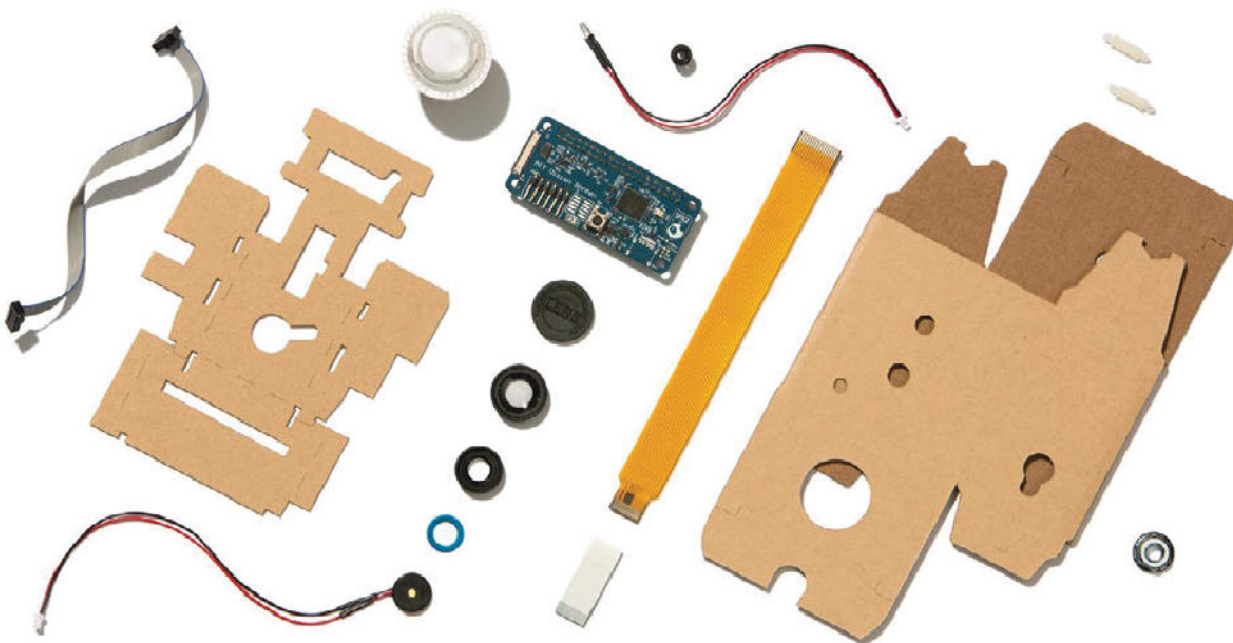
Il Pi Camera Module è connesso direttamente al Vision Kit bonnet

Movidius MA2450

Il cuore pulsante del Vision Bonnet, questo potente chip VPU rende il Pi Zero W 60 volte più veloce nel vision processing

Piedini GPIO

Il Vision Bonnet ha sei piedini GPIO in uscita, che ti consentono di integrare il kit con i tuoi progetti hardware

ORDINARLO

Il Vision Kit di AIY Projects sarà disponibile per il pre-ordine da Micro Center nel Dicembre 2017. La disponibilità iniziale sarà molto limitata. Se sei interessato, noi ti consigliamo di ordinare un kit al più presto, appena sarà disponibile. microcenter.com

mostrare ai maker che possono costruire da soli cose piuttosto impressionanti. Come nei precedenti AIY Projects, Google è interessata a vedere che idee vengono fuori dai maker con l'uso della IA per "risolvere i problemi per se stessi, le loro famiglie, e le loro comunità."

"Abbiamo sviluppato un motore a accelerazione interferenziale a profondo apprendimento, che stiamo facendo girare sul chip", spiega Kai Yick dal team AIY Projects. "È 60 volte più veloce rispetto a provare a farlo su un Raspberry Pi 3."

> Il Vision Kit aggiunge al RaspberryPi una avanzata scheda hardware di IA

Tutti a bordo della IA

The Vision Kit aggiunge una scheda hardware di intelligenza artificiale (IA) avanzata al Raspberry Pi. Sviluppata da Google, si chiama Vision Bonnet e monta un potente chip di vision processing Movidius MA2450 (**movidius.com/myriadx**).

Il chip agisce come un "acceleratore di rete neurale" afferma Billy Rutledge, direttore degli AIY Projects di Google. "Nel caso del Vision Kit, stiamo facendo un bel salto da gigante in avanti, facendo girare la rete neurale della IA sulla scheda accessoria stessa."

Questo è in contrasto con il precedente AIY Projects Voice Kit, che faceva affidamento sull'infrastruttura Cloud di Google per il riconoscimento vocale e l'elaborazione del linguaggio naturale.

I progetti che costruirai opereranno indipendentemente dalla connessione di rete, rendendolo un componente molto versatile. Il Vison Bonnet assicura anche la sicurezza delle immagini catturate, poiché vengono tutte elaborate localmente sul dispositivo.

Come è fatto il Kit

Una volta che hai assemblato il Vision Kit, ci sono un numero di programmi software di reti neurali che puoi lanciare. La prima rete neurale è un "rilevatore di persone, gatti e cani" rivela Peter Malkin, Software Lead agli AIY Projects. Essa rileva se una persona, un gatto o un cane è nell'inquadratura.

La seconda rete neurale è focalizzata sulle emozioni facciali. Rileverà felicità, tristezza e altri sentimenti.

La terza rete neurale può identificare 1,001 oggetti comuni, come una tazza, un'arancia o una sedia. Una label mostra il nome e il livello di fiducia nell'interferenza della rete neurale.

In futuro, la speranza è che gli utenti saranno in grado di modificare queste reti neurali. Potresti prendere il modello di gatto, cane e persona e modificarlo per cercare conigli. Poi costruire un progetto che funzioni come una trappola per conigli, per esempio. Sia *The MagPi* che *Google* non vedono l'ora di scoprire cosa hanno in serbo i maker per il Vision Kit di AIY Projects. I Pre-ordini per il Vision Kit inizieranno a dicembre presso Micro Center (**microcenter.com**).

Sotto Il Vision Kit è un progetto di smart camera utile per riconoscere facce, stati d'animo e oggetti; puoi costruirlo a casa tua e integrarlo nei tuoi progetti

INIZIARE CON L'ELETTRONICA

Fai pratica con i circuiti, i componenti e il physical computing con un computer Raspberry Pi

LISTA DEI MATERIALI

Una lista di materiali (o "BOM") è un elenco di componenti necessari per riprodurre un progetto. Tipicamente in un progetto Raspberry Pi, è un lista di componenti elettronici e software da installare.

Oltre ad essere il più bello, il più elegante, il miglior computer in circolazione, il Raspberry Pi anche dotato di pin GPIO. Questi piedini consentono al tuo Raspberry Pi di interagire fisicamente con il mondo esterno. Puoi collegarli a dei circuiti e aggiungere componenti elettronici come luci LED, cicalini, pulsanti, e sensori.

Ci sono 40 piedini in totale e non mancano i componenti che puoi usare. Questo è ciò che rende Raspberry Pi una parte così importante del mondo informatico.

Con questi piedini, puoi trasformare Raspberry Pi praticamente in qualsiasi cosa.

Questo speciale è per coloro che voglio sapere come costruire, hackerare e fare making con i computer. In questa guida, imparerai a leggere gli schemi, i circuiti prototipo, e a costruire kit elettronici di base.

Benvenuto nel meraviglioso mondo dell'elettronica con Raspberry Pi.

PERCHÉ IMPARARE L'ELETTRONICA DI BASE?

Vuoi essere solo un utente di gadget? O vuoi essere un maker, un hacker e un fixer? Qualcuno che sa cosa c'è sotto la superficie, chi sa come risolvere, adattare e migliorare il mondo che lo circonda.

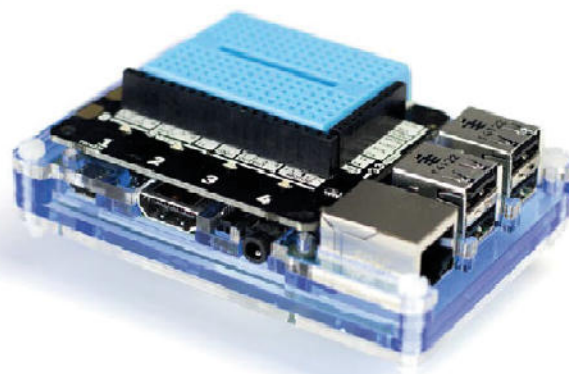
Raspberry Pi non è solo un computer economico e incredibilmente ben fatto. Ti aiuta a imparare come funzionano i computer.

Certo, c'è molto da fare. C'è da imparare a programmare e come usare strumenti da maker come stampanti 3D e il nastro americano. Ma fare prototipi dei propri circuiti circuiti e creare le tue schede elettroniche e imparare come collegare i componenti tra loro è una parte vitale del processo di making.

Il mondo moderno è infuso di dispositivi elettronici e il il futuro sarà costruito su dispositivi elettronici con intelligenza artificiale. Quindi è incredibilmente importante capire come funzionano queste cose. Imparare le basi della realizzazione di un circuito ti aiuterà a navigare il futuro.

> "Benvenuti nel fantastico Mondo dell'elettronica con Raspberry Pi"

COS'È UN HAT?

Gli HAT (Hardware Attached on Top) sono schede elettroniche pre-costruite con componenti vari. Sono progettati per connettersi al connettore a 40 pin del GPIO sul tuo Raspberry Pi, e sono facili da configurare e utilizzare. Gli HAT offrono un modo privo di problemi per usare l'elettronica, e alcuni (come il Pimoroni Explorer HAT, **magpi.cc/1OAKy46**) aggiungono una breadboard al tuo Raspberry Pi. Per questo speciale, eviteremo gli HAT e ci concentreremo sull'uso dell'elettronica pura, collegata direttamente ai pin GPIO di Raspberry Pi.

PROTOTIPAZIONE CON RASPBERRY PI

Ecco un tipico progetto con Raspberry Pi e breadboard. Dei fili, dal Raspberry Pi, sono collegati ai componenti del circuito. Sul Pi viene utilizzato del codice per controllare il circuito.

01. PIN GPIO

I pin GPIO su un Raspberry Pi vengono utilizzati per ottenere degli input dai componenti, come ad esempio un pulsante premuto o il valore di un sensore. Possono anche fornire un output e essere quindi accesi o spenti (portati a livello logico"ALTO" o 'BASSO') per controllare i componenti.

02. CAVALLOTTI

I cavallotti sono usati per collegare i piedini del GPIO di Raspberry Pi ai componenti sulla breadboard (e a collegare i componenti tra loro).

03. BREADBOARD

Una breadboard viene utilizzata per creare un prototipo di un circuito. I componenti sono inseriti nei fori su di essa, e per collegarli a ciascuno altro (e ai pin GPIO di Raspberry Pi) vengono usati i cavallotti.

04. COMPONENTI ELETTRONICI

Sono disponibili in tutte le forme e dimensioni ed eseguono una ampia varietà di compiti. Componenti comuni sono LED, resistenze, potenziometri, sensori PIR, e interruttori.

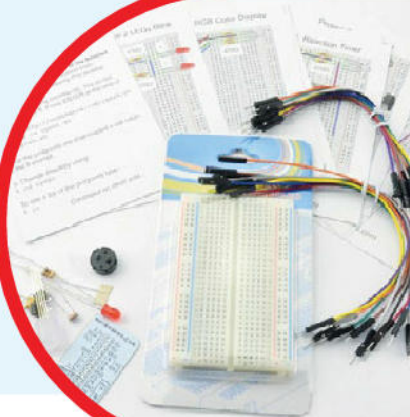
KIT

È piuttosto comune acquistare kit contenenti un certo numero di componenti. Questi possono essere utilizzati con Raspberry Pi resources e altre guide o tutorial

CAMJAM EDUKIT #1 E #2

Ci sono due pacchetti CamJam EduKit disponibili su The Pi Hut (magpi.cc/2yV1VQY). Il primo (circa 6 €) contiene una breadboard, dei LED, pulsanti e cicalini. Il secondo (circa 9 €), ha anche sensori di temperatura e PIR. Entrambi sono forniti di fili "cavallotti" e fogli di lavoro scaricabili da seguire per sperimentare con essi.

ELECTRONICS STARTER KIT PER RASPBERRY PI

Simon Monk è un popolare scrittore di guide si elettronica (ha scritto diverse guide per ***The MagPi***). Il suo starter kit (magpi.cc/2eC95jz, 11/19 €) ha ogni genere di componente e dieci progetti da realizzare.

RASPBERRY PI YOUTUBE WORKSHOP KIT

Questo kit da ModMyPi (magpi.cc/2mjJJdB, 18 €) contiene una breadboard trasparente (puoi quindi vedere le connessioni interne). Ha anche LED, resistenze, pulsanti, cicalini, sensori e fili a cavallotto. È uno dei kit più completi e viene fornito con una gamma completa di Video su YouTube per aiutarti lavorare con ogni componente (magpi.cc/2jj0ftf).

KIT UTILE

comodo kit per iniziare a imparare l'elettronica con un Raspberry Pi

BREADBOARD

Una breadboard è una basetta di plastica con un mucchio di buchi. A prima vista, sembra abbastanza incomprensibile, ma presto diventa semplice da capire. Vedi la nostra lezione Come utilizzare una breadboard (magpi.cc/2jB52Vz). Noi usiamo una breadboard Raspberry Pi (mezza taglia) nei nostri schemi (magpi.cc/2q34ZFz).

Contatti

Nella parte centrale della breadboard ci sono delle colonne di fori (tipicamente due), chiamate strip. Queste sono disposte generalmente in file da cinque, e ogni foro in un gruppo di cinque è collegato a quello successivo ad esso. I componenti inseriti in a una fila di cinque fori sono collegati tra di loro vicendevolmente. Quindi puoi connettere i componenti del circuito tra loro, senza doverli collegare fisicamente assieme.

Binari di alimentazione

Su ognuno dei lati della breadboard, troviamo due file di fori (che corrono lungo tutta la basetta). Queste sono detti binari positivo e negativo e sono tipicamente collegati a una batteria, o al pin positivo (3v3/5V) e massa (GND) sul Raspberry Pi.

Supporto DIP

Al centro di una breadboard, tra le due colonne di strip, vi è una scanalatura. Questo è solitamente lo spazio esatto per posizionare un chip DIP (dual in-line package). Questi chip CI possono sormontare la scanalatura centrale con una fila di piedini che finiscono nei fori su entrambi i lati.

CAVALLOTTI

I cavetti ponticello o cavallotti sono utilizzati per collegare i piedini del GPIOsu un Raspberry Pi ai componenti sulla tua breadboard. Alcuni cavallotti hanno terminali femmina che si connettono ai pin GPIO sul Raspberry Pi e terminali maschio che si connettono ai fori su una breadboard. In genere avrai bisogno di una selezione di cavallotti femmina-maschio e maschio-maschio magpi.cc/2muhzg4

LED

Se sei assolutamente nuovo all'elettronica su Raspberry Pi, compra una manciata di LED e resistenze (con breadboard e cavallotti). Ti consentiranno di creare un semplice circuito che accende un LED con il codice, come quello in questo tutorial online: Physical Computing with Python. magpi.cc/2mjGMjZ

RESISTENZE

I LED sono cose delicate. Se fai scorrere troppa corrente attraverso di loro, scoppieranno (a volte in modo abbastanza spettacolare). Per limitare la corrente che passa attraverso il LED, dovresti sempre usare una resistenza in serie ad esso. la resistenza può essere una qualsiasi oltre 50 Ohm circa. magpi.cc/2mtKKjs

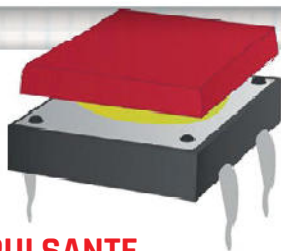
Per leggere il valore di una resistenza, leggi le sue fasce colorate. Una delle estremità avrà una striscia sopra, tipicamente argento o oro. Assicurati che sia sulla destra. Adesso guarda da vicino le bande nel mezzo, le prime due bande rappresentano numeri e il il terzo è un moltiplicatore. Nella resistenza qui sotto, abbiamo due bande arancioni (3) e un moltiplicatore marrone (10). Quindi il nostro
la resistenza è 33 × 10 o 330 ohm). La fascia d'oro finale indica che il valore della resistenza ha una tolleranza, o precisione. La nostra è del 5%.

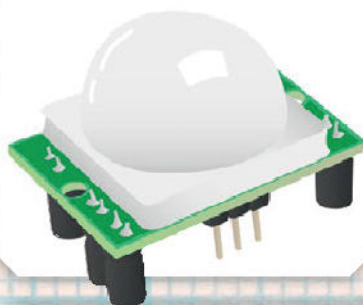
Fascia 1: Numero	Fascia 2: Numero	Fascia 3: Moltiplicatore	Fascia 4: Tolleranza
NERO: 0	NERO: 0	NERO: 1	MARRONE: 1%
MARRONE: 1	MARRONE: 1	MARRONE: 10	ROSSO: 2%
ROSSO: 2	ROSSO: 2	ROSSO: 100	ORO: 5%
ARANCIO: 3	ARANCIO: 3	ARANCIO: 1000	ARGENTO: 10%
GIALLO: 4	GIALLO: 4	GIALLO: 10000	NESSUNA: 20%
VERDE: 5	VERDE: 5	VERDE: 100000	
BLU: 6	BLU: 6	BLU: 1000000	
VIOLA: 7	VIOLA: 7	ORO: 0.1	
GRIGIO: 8	GRIGIO: 8	ARGENTO: 0.01	
BIANCO: 9	BIANCO: 9		

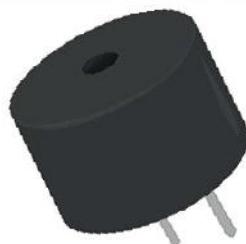
CONNETTORE "9V"

Le uscite di Raspberry Pi, da alcuni dei suoi pin GPIO, forniscono 3,3 V costanti ed è perfettamente possibile ottenerli senza una batteria. Ma noi troviamo che una batteria da 9V con un semplice cavetto, sia un modo pratico per aggiungere alimentazione a un circuito e testarne così i componenti.
magpi.cc/2mshkSG

PULSANTE

Un pulsante è un ottimo modo per provare l'input su un Raspberry Pi (dove crei del codice che risponde alla pressione del pulsante fisico). Alcuni pulsanti hanno due piedini; altri – come questo – ne hanno quattro, e sono progettati per adattarsi allo spazio per i DIP al centro di una breadboard.
magpi.cc/2mqTgj2

SENSORE PIR

Un sensore PIR (infrarosso passivo) è usato per rilevare il movimento. Come i pulsanti, sono un buon modo per aggiungere degli input a un progetto. Puoi creare uno script che rileva il movimento e risponde di conseguenza.
magpi.cc/2msRLkJ

CICALINO PIEZOELETTRICO

Questo cicalino è usato per aggiungere del suono base a un progetto. Sebbene non vada a scuotere la casa, un altoparlante piezoelettrico è un modo pratico per ottenere feedback da un progetto.
magpi.cc/2muqIFy

SENSORE DI DISTANZA A ULTRASUONI

Il sensore di distanza a ultrasuoni può determinare quanto è lontano un oggetto solido. Funziona inviando una serie di ultrasuoni. Questo suono viaggerà attraverso l'aria ma verrà riflesso indietro (eco) dalle superfici solide. Il sensore rileva l'eco. Si trova spesso sui robot e su altri progetti Raspberry Pi.
magpi.cc/2mtgpS1

SCHEMI ELETTRICI E DI CABLAGGIO

Ora che hai un mucchio di componenti, è il momento di iniziare a metterli insieme e imparare a programmare

Una volta che hai una breadboard e alcuni componenti elettronici, ti consigliamo di iniziare a collegarli e imparare come funzionano con Raspberry Pi.

Non c'è assolutamente mancanza di progetti elettronici là fuori, e molti di loro ti guidano attraverso il collegamento dei componenti alla tua breadboard, usando poi software come la libreria Python GPIO Zero.

Ci sono due approcci per descrivere il tracciato dei circuiti elettrici. Il primo è usare uno schema di cablaggio (vedi 'Schema di cablaggio').
Il semplice circuito mostrato di seguito collega un pulsante e un LED a piedini GPIO differenti sul Raspberry Pi. Il Raspberry Pi rileva la pressione del pulsante e accende il LED.

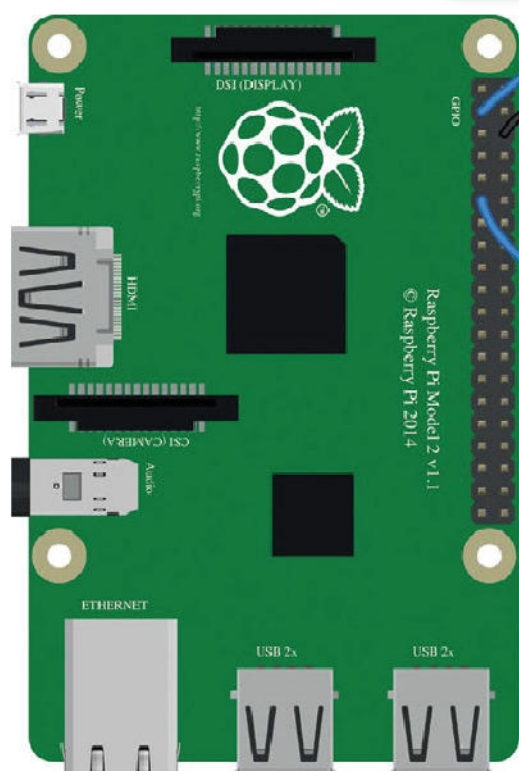
Troverai schemi di cablaggio come questo in *The MagPi*, e altri simili sono usati dalla Raspberry Pi Foundation e altre risorse. Se sei interessato a creare il tuo, usa un programma chiamato Fritzing per realizzarlo (**fritzing.org**).
Se vuoi una guida per i piedini GPIO e un corso accelerato su come controllarli con GPIO Zero, dai un'occhiata alla nostra Guida per Principianti - Guida a GPIO Zero su The MagPi numero 52 (**magpi.cc/Issue-52**).

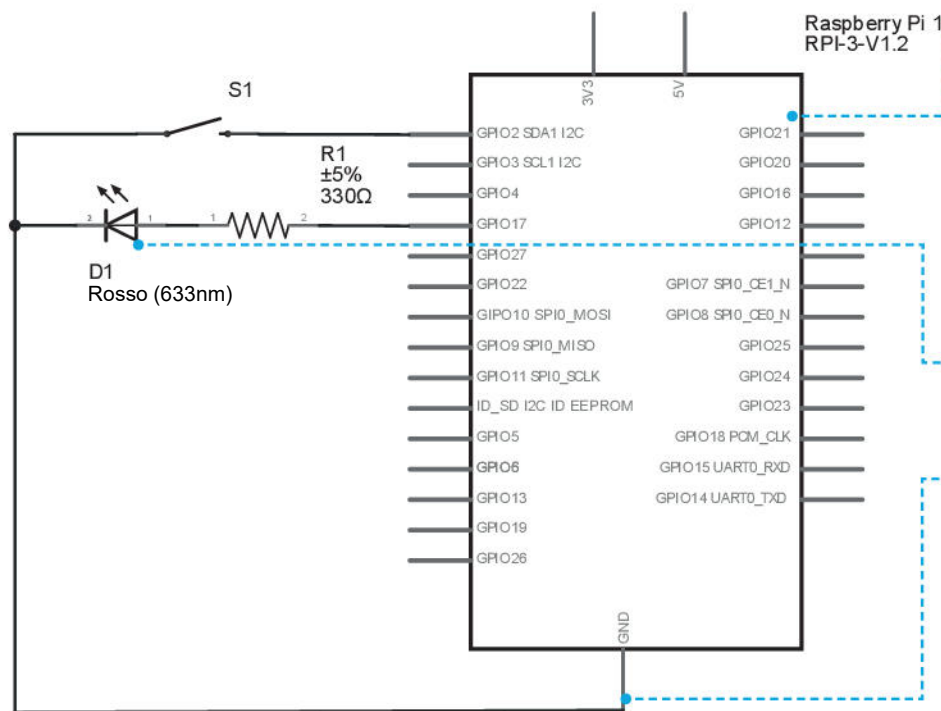
SIMPLE ELECTRONICS WITH GPIO ZERO

Per maggiori informazioni sull'utilizzo di GPIO Zero e per imparare a collegare i circuiti dallo schema di cablaggio. dai uno sguardo alla nostra Essentials Guide su GPIO Zero. magpi.cc/2bA3ZP7

SCHEMA DI CABLAGGIO

Fili
I cavallotti sono rappresentati da questi fili colorati

Breadboard
Segui le linee di fori della breadboard per vedere come i componenti sono collegati

I componenti
I componenti sono visivamente simili alle loro controparti reali.. Alcuni, come il LED, mostrano visivamente in quale verso dovrebbero andare

SCHEMA ELETTRICO

I pin GPIO di un Raspberry Pi sono rappresentati da questo grande rettangolo. Non sono disegnati nella stessa posizione come i piedini fisici sulla scheda elettronica reale

I componenti sono rappresentati con dei simboli, come questo triangolo con le frecce (che rappresenta un LED)

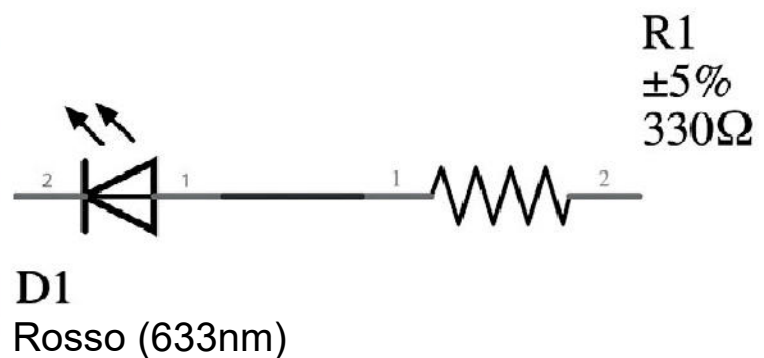
Vengono usate delle linee nere per indicare le connessioni tra i componenti sul circuito. Possono rappresentare dei fili o componenti collegati allo stesso binario su una breadboard

COME CAPIRE GLI SCHEMI ELETTRICI

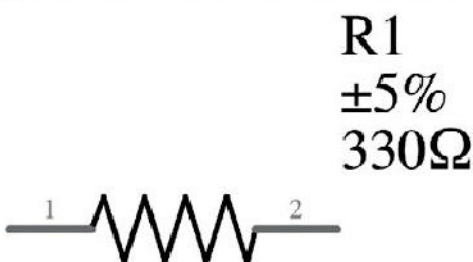
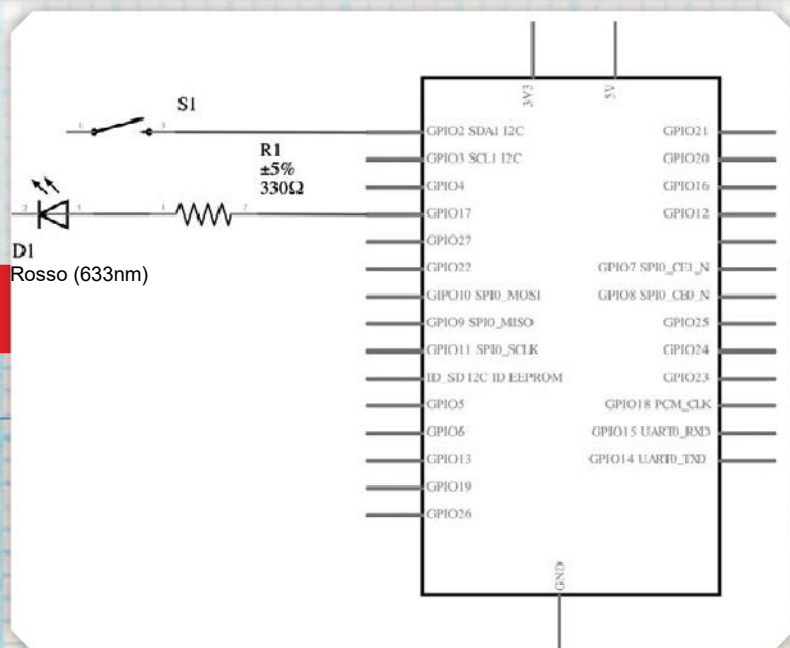
>PASSO-01

Componenti

Gli schemi elettrici descrivono come i componenti sono collegati tra loro. A differenza degli schemi di cablaggio, i simboli non sono visivamente simili ai componenti del mondo reale e il e il simbolo non necessariamente somiglia alla loro forma. È una guida su come sono collegate le cose.

Prendi la resistenza. Invece di essere visivamente realistica, cioè un tubo con anelli colorati, è indicata da queste linee a zigzag (all'estero, a volte invece di questo simbolo, si usa un rettangolo). Le caratteristiche sono spesso elencate con del testo accanto. Imparare a identificare i componenti è metà del lavoro. Vedi la alla 'Guida di riferimento ai componenti' (a pagina 25), che confronta i simboli con le loro controparti del mondo reale.

>PASSO-02

Collegamenti

Ora abbiamo aggiunto un secondo componente alla nostra resistenza: un LED. È rappresentato da un triangolo (il simbolo di un diodo, che indica che il flusso di corrente può attraversarlo solo in una direzione). Il simbolo del LED ha anche due frecce rivolte verso l'esterno, che rappresentano la luce emessa. Uniamo i due componenti assieme con una linea nera. Questa indica che sono collegati (sulla nostra breadboard di prototipazione potrebbero essere nella stessa fila di buchi, o collegati con un ponticello o cavallotto).

>PASSO-03

Circuiti integrati

Il nostro Raspberry Pi è visualizzato come un CI (Circuito Integrato). Questo grande rettangolo ha piedini tutt'attorno che rappresentano i piedini del connettore GPIO. Noi abbiamo collegato i nostri resistenza e LED al GPIO 17 (come scritto dentro al rettangolo del CI). Come gli altri componenti, il CI schematico è una astrazione e non assomiglia alla disposizione fisica dei piedini. Utilizza uno schema dei pin (it.pinout.xyz) per individuare la posizione dei piedini fisici sulla scheda del tuo Raspberry Pi.

>PASSO-04

Fili e nodi

A volte, i fili formano una giunzione che connette più componenti insieme. Il nostro pulsante è su un piedino del GPIO separato rispetto al LED e la resistenza, ma entrambi si connettono insieme e vanno a collegarsi a un piedino di massa singolo. I fili che si collegano a incrocio sono indicati da grandi cerchi chiamati 'nodi'. Quando vedi questo cerchio, significa che i fili nel circuito si connettono in questo punto.

>PASSO-05

Fili incrociati

In uno schema ben progettato, le varie parti saranno mantenute separate e i fili non si incroceranno tra loro. Questo non è sempre possibile con schemi più complessi, e a volte i progetti non sono chiari come potrebbero essere. Se vedi due fili che si incrociano senza un il cerchiolino del nodo, significa che i fili stanno solo "passando" (e non sono collegati).

>PASSO-06

Nomenclatura di riferimento

Ora che sai come identificare i fili e i componenti sullo schema, hai semplicemente la necessità di identificare ciascun componente e l'ordine in cui sono collegati. I nomi dei componenti sono normalmente composti da una lettera e un numero. Quindi la tua prima resistenza è R1, la seconda R2 e così via. È possibile visualizzare un elenco di tutti i nomi di riferimento più comuni su Wikipedia (**magpi.cc/2mqvyUf**). Buona prototipazione!

GUIDA DI RIFERIMENTO COMPONENTI

Ecco alcuni componenti comuni assieme alla loro lettera di nomenclatura di riferimento, la grafica di cablaggio e il simbolo negli schemi elettrici.

Componente	Nomenclatura di riferimento	immagine nello schema di cablaggio	Simbolo schematico
Resistenza	R		
Condensatore	C		
Diodo (LED)	D		Anode Cathode
Interruttore	S		
Transistor	Q		E B C

Schematic symbols courtesy of Wikipeda.org

REALIZZA UNA GPIO MUSIC BOX

Questo progetto ti mostra come collegare i pulsanti ai piedini del GPIO di Raspberry Pi e poi usarli per riprodurre suoni con una semplice applicazione Python

COSA SERVE

- Raspberry Pi
- Breadboard
- 4 interruttori tattili per fare i pulsanti
- 5 Cavallotti di collegamento maschio-femmina
- 4 cavallotti maschio-maschio

In questo tutorial, useremo più pulsanti per creare una GPIO music box che emette suoni differenti quando premiamo diversi pulsanti.

Per fare questo, useremo la classe **Button** di GPIO Zero per assegnare campioni sonori ai pulsanti. Anche se abbiamo fornito uno schema di cablaggio, ti suggeriamo di provare a seguire per primo lo schema elettrico, usando le conoscenze che hai appena acquisito. Se questo diventa un problema, confronta lo schema di cablaggio con lo schema elettrico.

>PASSO-01

Ottieni qualche suono

Prima di iniziare a costruire il nostro circuito musicale col GPIO, dovremo preparare alcuni campioni sonori per farlo suonare. Innanzitutto, apri una finestra del Terminale e crea un directory chiamata **musicbox** per questo progetto. poi crea una directory chiamata **samples** dentro a **musicbox**.

Abbiamo necessità di trovare alcuni campioni sonori. Anche se esistono on-line molti suoni di pubblico dominio, per questo esempio useremo alcuni dei suoni di percussioni integrati in Scratch, che sono già presenti sul Pi. Copia i suoni di percussione di Scratch nella directory **samples**.

```
mkdir musicbox
mkdir musicbox/samples
cp /usr/share/scratch/Media/Sounds/
Percussion/* /home/pi/musicbox/samples
```

>PASSO-02

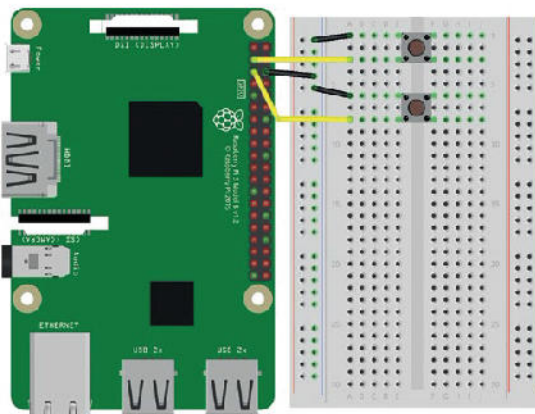
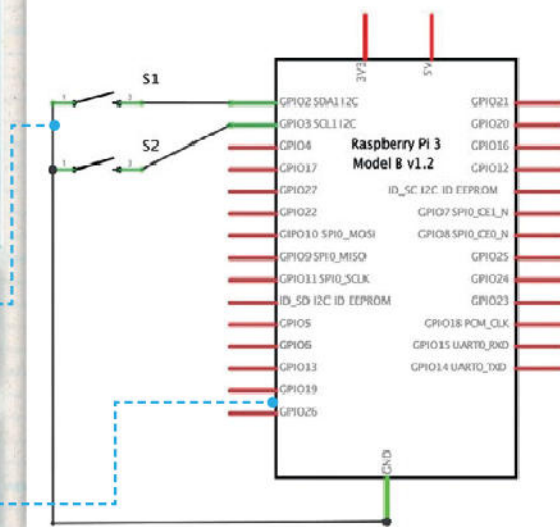
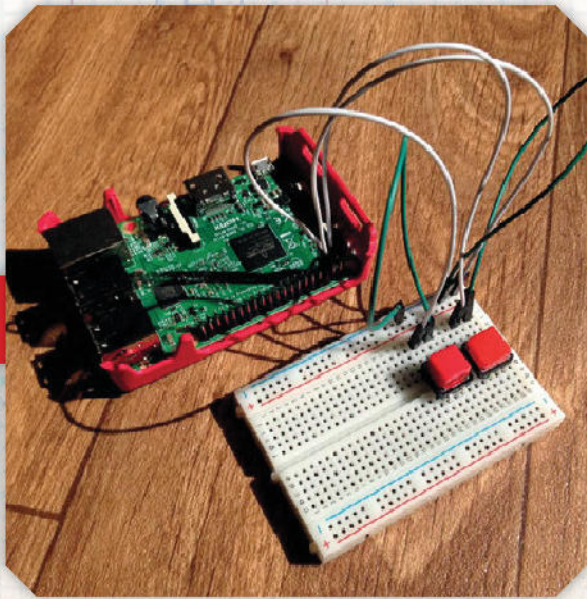
Aggiungi un pulsante

Ora segui lo schema e collega il primo pulsante. Posiziona il pulsante in modo che si trovi a cavallo

Ricorda, i pulsanti non sono collegati a niente altro. Questo punto di giunzione indica che entrambi sono collegati a un singolo piedino di massa

Le posizioni dei piedini del GPIO sullo schema di cablaggio non corrispondono a quelli sulla scheda fisica. Usa it.pinout.xyz se hai bisogno di una guida sulla posizione dei piedini

SCHEMA DI CABLAGGIO

Prova a collegare il circuito usando lo schema sotto. Se hai difficoltà a seguirlo, controlla questo schema di cablaggio. Altre istruzioni sono sul sito web risorse Pi (magpi.cc/2z44wrC).

SCHEMA GPIO MUSIC BOX

In alto Ogni volta che si preme un pulsante, il campione sonoro assegnato verrà suonato tramite un altoparlante collegato

A destra Dei pulsanti extra possono essere facilmente aggiunti al circuito per riprodurre più suoni assegnati nel codice Python

della scanalatura centrale della breadboard. Un pin è collegato al piedino GPIO 2 e l'altro al binario di massa della breadboard, che a sua volta è collegato a un pin GND.

>PASSO-03

Aggiungi un secondo pulsante

Ora aggiungi un secondo pulsante al circuito, ancora seguendo lo schema. Mettilo sulla breadboard come prima, e collegalo al GPIO 3 e al binario comune di massa.

Ora apri un nuovo file nell'IDLE di Python 3, inserisci il codice da **gpio_musicbox.py** e salvalo nella tua cartella **musicbox**. Esegui il programma e premi i pulsanti. Dovresti sentire un rumore quando ogni pulsante viene premuto.

>PASSO-04

Aggiungi altri pulsanti

Il modo in cui abbiamo strutturato il programma, rende facile aggiungere pulsanti extra e assegnarli a dei suoni campionati. Basta collegare ciascun pulsante a un piedino 'GPIO numero' (non di qualsiasi altro tipo) e al binario di massa, come prima.

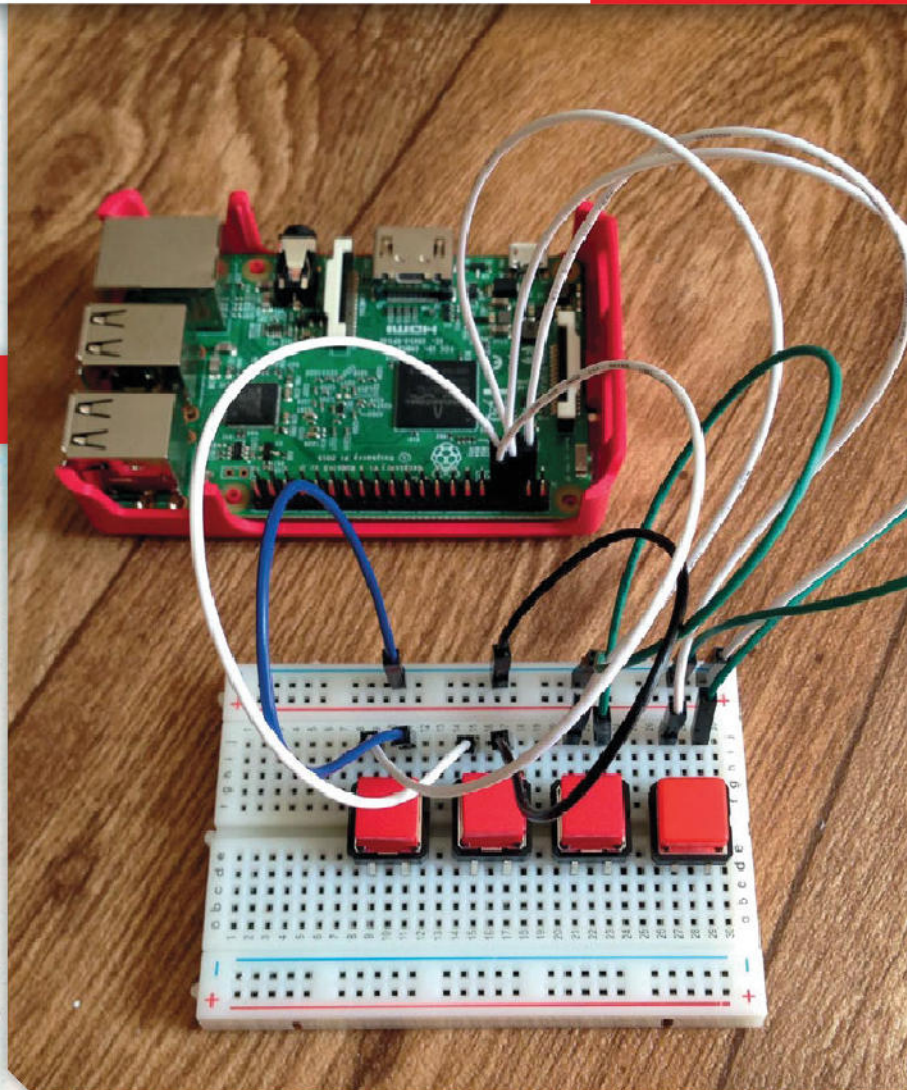
Quindi aggiungi i numeri e i suoni dei piedini GPIO al dizionario, come nell'esempio seguente:

```
sound_pins = {
    2: Sound("samples/DrumBizz.wav"),
    3: Sound("samples/CymbalCrash.wav"),
    4: Sound("samples/Gong.wav"),
    14: Sound("samples/HandClap.wav"),
}
```

Aggiungi altri pulsanti e ridisegna lo schema con i pulsanti collegati al CI Raspberry Pi.

gpio_musicbox.py

Language
>PYTHON

DOWNLOAD:
magpi.cc/2BaGLLM

```
import pygame.mixer
from pygame.mixer import Sound
from gpiozero import Button
from signal import pause

pygame.mixer.init()

button_sounds = {
    Button(2): Sound("samples/drum_tom_mid_hard.wav"),
    Button(3): Sound("samples/drum_cymbal_open.wav"),
    Button(4): Sound("samples/elec_bell.wav"),
    Button(14): Sound("samples/elec_hi_snare.wav"),
}

for button, sound in button_sounds.items():
    button.when_pressed = sound.play

pause()
```

REALIZZA UN TELEMETRO

Collega tra loro un sensore a ultrasuoni e un display a sette segmenti per misurare le distanze

COSA SERVE

- Sensore di distanza a ultrasuoni HC-SR04 magpi.cc/2mtgpS1
- Display a sette segmenti Broadcom 5082-7650 magpi.cc/1YnqQPl
- 9 Resistenze (7 × 220 Ω, 1 × 512 Ω, 1 × 1 kΩ) magpi.cc/1YnqU1r

Il sensore di distanza ultrasonico HC-SR04 è un grande favorito tra i produttori di robot Pi. Funziona facendo rimbalzare il suono a ultrasuoni contro un oggetto e cronometrando quanto tempo ci vuole avere indietro l'eco. Questo tempo viene quindi convertito in una distanza che può essere visualizzata su un display a sette segmenti a una cifra. Qui puoi acquisire le abilità per gestire gli input e gli output. Comincerai anche a prendere pratica con i display a sette segmenti, che sono abbastanza fighi, in uno stile retrò.

>PASSO-01

Accendere il display

Il display a sette segmenti è un insieme di LED, dove ogni LED corrispondente a uno dei segmenti. Tutti gli anodi (terminali positivi) sono collegati assieme; questo dovrebbe essere collegato al positivo di 3,3 V (3V3 pin).
I catodi (terminali negativi) devono essere collegati a una resistenza per limitare la corrente che attraversa il LED e l'altra estremità della resistenza, a un pin GPIO. Per accendere il LED, tutto ciò che devi fare è impostare l'uscita GPIO come livello BASSO (0V) e questo completerà il circuito per far fluire la corrente.

Puoi utilizzare qualsiasi display a sette segmenti, ma altri potrebbero avere un pinout differente

Sensore di distanza ad ultrasuoni: fornisce un impulso proporzionale a qualsiasi oggetto riflettente davanti ad esso

Il sensore di distanza ad ultrasuoni HC-SR04 viene visualizzato come un CI con quattro piedini

>PASSO-02

Generare un modello a sette segmenti

Il display consiste di sette barre o segmenti che possono essere accesi. Scegliendo i segmenti da illuminare, puoi mostrare un numero da 0 a 15, anche se devi ricorrere alle lettere per questo (noti anche come numeri esadecimali). Vi sono, infatti, 128 modelli diversi possibili, ma la maggior parte di essi, è priva di senso. Nel nostro codice (**range_finder.py**), una lista chiamata **seg** definisce quali pin sono connessi a quale segmento, e un'altra lista, chiamata **segmentPattern** definisce il modello LED per ogni numero.

Ricorda che due fili che si attraversano non sono collegati a meno che non ci sia un grande nodo circolare. In questo schema, le linee sono colorate in modo diverso per aiutare a fare lo "sbroglio" dei fili

In alto Il progetto in azione; il Pi sta misurando quanto è distante dalla confezione di Raspberry Pi 3

>PASSO-03
Mostrare numeri

La funzione **display** imposta i segmenti e visualizza qualsiasi numero a una cifra che gli viene passato. Prima imposta tutti i segmenti su off e quindi se il numero è inferiore a 16, passa attraverso gli elementi della lista **segmentPattern** cercando quel numero e attivando i segmenti appropriati. Nota che possiamo ancora usare **on** e **off** anche se non sono alimentati da singoli pin GPIO, come i LED erano dichiarati a GPIO Zero come **active_high = False**.

>PASSO-04
Il sensore di distanza

Il sensore di distanza HC-SR04 trasmette la sua lettura producendo degli impulsi in uscita che il Pi prova a misurare. La libreria GPIO Zero misura questi impulsi e li converte in una distanza restituendo un numero in virgola mobile che arriva a un massimo di 1 metro. Quindi moltiplichiamo questo numero per 10 per avere i decimetri. Successivamente, lo convertiamo in un numero intero per sbarazzarci della parte frazionaria della misura, quindi possiamo mostrarlo sul nostro display a una sola cifra.

range_finder.py

Language
>PYTHON

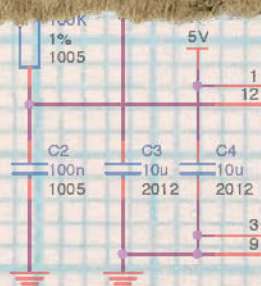
DOWNLOAD:
magpi.cc/2BaGLLM

```
01. # mostra la distanza in decimetri su un display a
    7 segmenti
02. from gpiozero import LED
03. from gpiozero import DistanceSensor
04. import time
05.
06. seg = [LED(27,active_high=False),LED(25,active_
    high=False),LED(24,active_high=False),
07.       LED(23,active_high=False),LED(22,active_
    high=False),LED(18,active_high=False),
08.       LED(17,active_high=False)]
09.
10. segmentPattern = [[0,1,2,3,4,5],[1,2],[0,1,6,4,3],[0,1
    ,2,3,6],[1,2,5,6],[0,2,3,5,6], #da 0 a 5
11.                  [0,2,3,4,5,6],[0,1,2],[0,1,2,3,4,5,6]
    ,[0,1,2,5,6],[0,1,2,4,5,6], #da 6 a A
12.                  [2,3,4,5,6],[0,3,4,5],[1,2,3,4,6],[0,
    3,4,5,6],[0,4,5,6] ] #da B a F
13.
14. sensor = DistanceSensor(15,4)
15.
16. def main() :
17.    print("Display distance on a 7-seg display")
18.
19.    while 1:
20.        distance = sensor.distance * 10 # distanza in
    decimetri
21.        print("distance",distance)
22.        if distance >= 10.0:
23.           distance = 16.0
24.        display(int(distance))
25.        time.sleep(0.8)
26.
27. def display(number):
28.     for i in range(0,7):
29.        seg[i].off()
30.     if number < 16:
31.        for i in range(0,len(segmentPattern[number])):
32.            seg[segmentPattern[number][i]].on()
33.
34. # Logica main program:
35. if __name__ == '__main__':
36.     main()
```

>PASSO-05
Costruire il progetto

Per questa realizzazione, abbiamo usato una piccola breadboard dinky shield di Dtronix. Questo ha permesso una disposizione molto più compatta di un convenzionale breadboard, anche se ovviamente tu la puoi utilizzare comunque. Poiché l'HC-SR04 utilizza un'alimentazione da 5 V, gli impulsi che dobbiamo misurare sono anch'essi a 5V nominali. Pertanto, devono essere ridotti a 3,3 V usando un partitore di tensione con resistenze da 512 Ω e 1 kΩ.

>PASSO-06
Usare il sensore

La distanza dall'oggetto riflettente viene aggiornata ogni 0,8 secondi. Se questa è maggiore di un metro, allora il il display sarà vuoto. Un display con 0 indica che il l'oggetto è a meno di 10 cm di distanza. Non toccare il sensore, altrimenti le sue letture saranno errate. Inoltre, siccome ha un raggio piuttosto ampio, è possibile ottenere riflessi dai lati. Se diversi oggetti sono nel campo visivo, allora viene restituita la distanza dal più vicino.

CREA LA TUA AVVENTURA DIGITALE GAMEBOOK con Twine

Utilizzando Twine 2, chiunque può creare facilmente un gioco interattivo basato su scelte e pubblicarlo online

K.G. ORPHANIDES

K.G. è un giornalista tecnologico i cui esperimenti in narrativa interattiva hanno spaziato da Multi-User Shared Hallucinations a gamebook e avventure testuali tradizionali.
twitter.com/KGOrphanides

Cosa Serve

- twinejs magpi.cc/2zpogTD
- Chromium
- Immaginazione, e forse qualche pezzo di carta

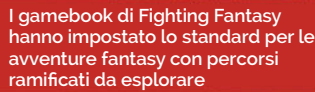
I giochi testuali hanno un retaggio quasi lungo altrettanto quanto l'informatica moderna. Creato su un mainframe PDP-10 nel 1975, Will Crowther's Colossal Cave Adventure è generalmente riconosciuto come la prima avventura testuale, permettendo ai giocatori di inserire semplici comandi come VAI A NORD e PRENDI LAMPADA.

Ha ispirato l'intero genere delle avventure testuali, con Infocom's Zork (1979) e Adventure International's Adventureland (1978) che diventarono tra le prime avventure testuali commerciali basate su parser.

Nel frattempo, un diverso tipo di narrativa interattiva Stava arrivando nelle librerie. Nel Regno Unito, i co-fondatori di Games Workshop Steve Jackson e Ian Livingstone hanno aggiunto degli elementi dei gioco di ruolo da tavolo ai loro videogiochi di Fighting Fantasy, il primo dei quali, *The Warlock of Firetop Mountain*, è stato pubblicato da Puffin Books nel 1982. Con passaggi più brevi, più scelte, una scheda personaggi e un dado sistema di combattimento, Fighting Fantasy sarebbe stato, per definizione, il gamebook di avventura per intere generazioni di bambini britannici.

Sono queste avventure che prenderemo di ispirazione per questo tutorial, mostrandoti come implementare non solo una ramificazione, basata sulla trama a scelte, ma anche eccitante azione, lanci fortunati dei dadi e dei personaggi che possono essere influenzati - e persino uccisi - dalle sfide che affrontano.

I gamebook di Fighting Fantasy hanno impostato lo standard per le avventure fantasy con percorsi ramificati da esplorare

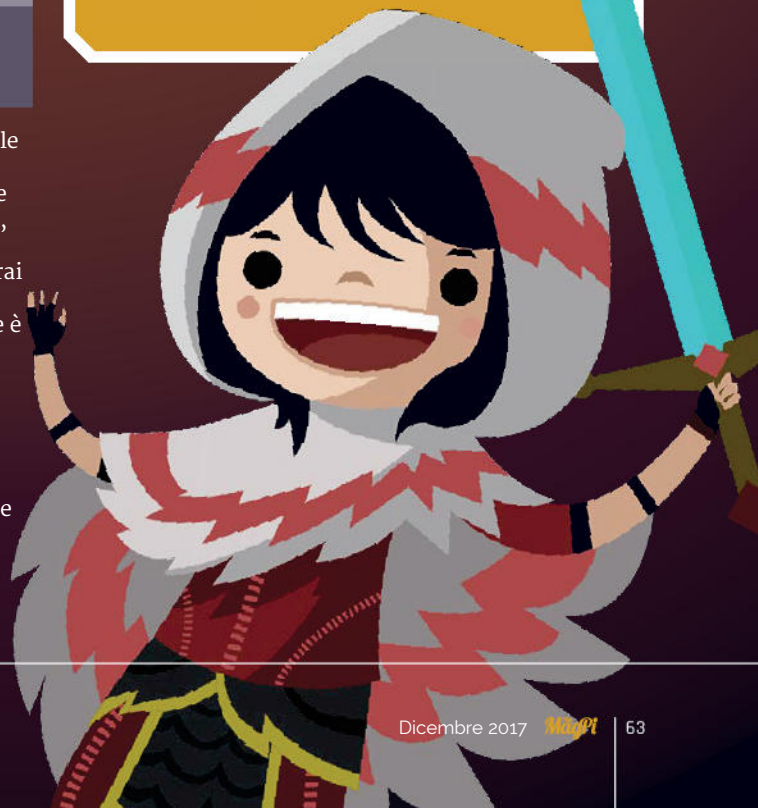
COSA È TWINE

In questo tutorial, useremo Twine 2 e Harlowe, il suo linguaggio di programmazione predefinito, per creare una avventura interattiva fantasy. Twine è appositamente progettato per essere facile da usare. È possibile creare una storia basata su scelte multiple in modo semplice, senza scrivere una singola riga di codice, ma offre anche un ambiente di programmazione sorprendentemente flessibile, che puoi usare per implementare logica, statistiche, registrazione di eventi e lanci di dadi casuali.

INSTALLARE E LANCIARE TWINE 2

Dei client stand-alone di Twine desktop sono disponibili per diversi sistemi, ma non per computer basati su ARM come il Raspberry Pi - per ora, ad ogni modo. Tuttavia, esiste una versione web locale del motore chiamato TwineJS mantenuto dal creatore originale Chris Klimas, e tutto ciò che serve per eseguirla è un browser web.

1. **Vai su magpi.cc/2zpogTD e scarica l'ultima versione di Twine – il file zip liscio, quello senza sistema operativo menzionato. In questo momento, è twine_2.1.3.zip.**
2. **Scompattalo nella sua directory.**
3. **Apri index.html per lanciare Twine.**

Gira in un browser, ma tutti i file sono memorizzati localmente. Il tuo gioco può essere modificato e costruito completamente offline, ma a differenza di altre implementazioni del client, dovrai esportare manualmente i tuoi lavori in corso d'opera (che è in realtà la migliore pratica per tutte le iterazioni di Twine).

Sebbene Twine 2 dovrebbe funzionare perfettamente in qualsiasi browser con JavaScript abilitato, lo eseguiremo su Chromium e ti suggeriamo di fare lo stesso mentre segui questo tutorial.

THE ROAD TO ADVENTURE

Per seguire questo tutorial e illustrare le tecniche utilizzate in esso, abbiamo creato un gioco demo: The Road to Adventure™, che puoi reperire online su magpi.cc/2ydf107.

Per importarlo nella tua copia di Twine, scarica il file sorgente HTML collegato alla pagina di gioco principale, apri Twine e seleziona "Import from file" nella barra di destra della schermata home di Twine.

IL TUO Primo GIOCO

La prima volta che lanci Twine 2, alcuni paragrafi tutorial ti indirizzano verso della documentazione online e ti spiegano come salvare il tuo lavoro. Dopodiché, sarai portato all'indice principale di Twine, dove possiamo iniziare a scrivere il nostro gioco.

PARTIAMO

Fai clic sul pulsante verde Story nella barra a destra e dai un nome al tuo racconto, quando ti viene richiesto. Un grafico simile a fogli di carta apparirà sullo schermo, con una singola casella nel mezzo, nominata, per ora, Untitled Passage. Un doppio clic su di essa, apre la schermata di modifica.

Le caselle di passaggio sono dove passerai la maggior parte del tuo tempo in Twine. Clicca sulla barra del titolo - attualmente contrassegnata come Untitled Passage - e dagli un nome. Solo tu sarai in grado di vedere il titolo, puoi quindi chiamarla in qualsiasi modo ti aggrada, in quanto non sarà visibile al giocatore.

Sotto, c'è il riquadro principale, dove inserirai il testo che i tuoi giocatori leggeranno. Come suggerisce, fai doppio clic sul testo di esempio attualmente presente e inizia a digitare per comporre la tua storia.

C'è anche una casella Tag, invisibile al giocatore, che puoi usare per contrassegnare i passaggi per un tuo riferimento o per applicare del testo ripetuto o del codice al tuo gioco, per esempio nelle intestazioni e piè di pagine dei passaggi. Siccome questo va oltre lo scopo del nostro tutorial, puoi imparare di più a riguardo, nella documentazione di Twine.

Tutto quel che scrivi in un passaggio viene automaticamente salvato nella cache del tuo browser in tempo reale. Questo significa che, anche se il tuo computer si blocca, nulla andrà perduto. Tuttavia, è molto importante eseguire il backup del tuo gioco (vedi 'Salva spesso') prima di fare qualsiasi modifica importante al suo testo o codice.

IL TUO PRIMO LINK

Nel nostro primo passaggio – chiamato Spitz Adventurers' Guild Barracks nel nostro gioco demo - abbiamo descritto la caserma della corporazione di avventurieri e impostato la trama per la nostra storia. La motivazione del nostro personaggio è semplice: cacciare i mostri per la fama

RISORSE

Twinery
twinery.org

Manuale Harlowe
twine2.neocities.org

Pubblica gratis su Itch.io
itch.io

Un collegamento al futuro

```
(link: "Go north to town")[(goto: "Town gate")]
[[Go east to the pass->Snow-filled pass]]
[[Go south to the forest->Into the forest]]
[[Go west to the cemetery->Cemetery gates]]
[[Visit the tavern]]
```

e la fortuna. Ma ora stiamo per portarli alla fase successiva della loro avventura.

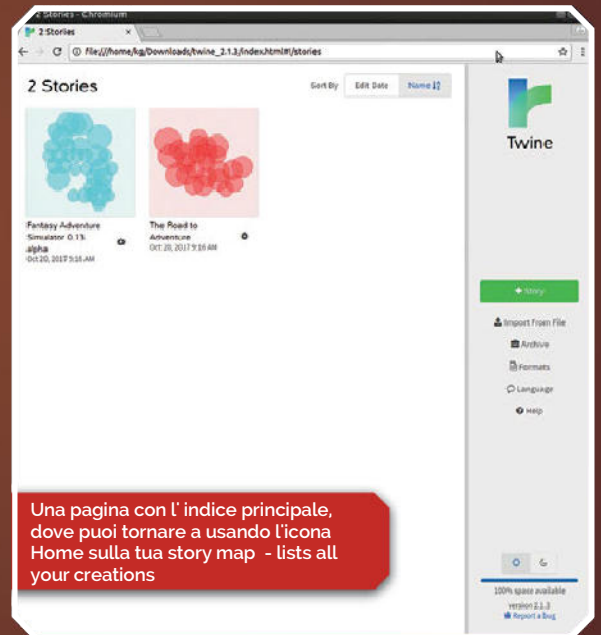
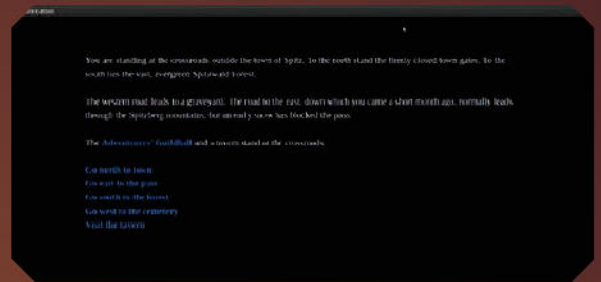
Per farlo, creeremo un semplice link. Quando si usa Twine e Harlowe, qualsiasi stringa di testo che metti tra due serie di parentesi quadre **[[come questo]]**, Twine creerà automaticamente un passaggio con lo stesso nome del tuo link. Fatto ciò, fai clic sulla X in alto a destra per chiudere.

CREARE STANZE, CREARE STANZE

Torna sulla schermata della mappa, vedrai che è stato creato un nuovo passaggio. Se lo rinominiamo, il link che punta ad esso sarà automaticamente aggiornato. Però, puoi anche creare un link che punta a un passaggio con un nome diverso dal testo del link.
Nel nostro codice di esempio (vedi 'Un collegamento al futuro') abbiamo usato diversi metodi di collegamento, uno dei quali comporta lo stesso formato tra parentesi che abbiamo usato prima, ma con una freccia che indica il nuovo nome del passaggio
[[così questo-> porta a questo]].

Un terzo metodo riguarda la nostra prima introduzione alla costruzione di una macro Harlowe. La macro **(link:)** crea un link che apparirà sulla nostra pagina corrente. Quando il giocatore clicca su di esso, il gioco manderà in uscita fuori qualsiasi comando all'interno di una coppia di parentesi.
In questo caso, questo è solo una macro (goto:), che farà lo stesso lavoro di un normale link. Tuttavia, non crea automaticamente il tuo passaggio di destinazione e non visualizzerà una freccia di collegamento sulla visuale di mappa.

Ciò significa che se creiamo un passaggio con questo po' di codice dentro – **(link: "Vai a nord") [(vai a: "Porta della città")]** – dobbiamo poi creare manualmente un nuovo passaggio chiamato porta della città, perché non sarà creato automaticamente quando usiamo questo metodo.

Generalmente, useremo solo questo tipo di collegamento, quando abbiamo bisogno di chiamare altre funzioni quando viene cliccato. Poi, useremo **(link:)** per dire a Twine di impostare le variabili, verificare gli stati precedentemente impostati ed eseguire il lancio dei dadi mentre ci spostiamo da un passaggio a quello successivo.

Una pagina con l' indice principale, dove puoi tornare a usando l'icona Home sulla tua story map - lists all your creations

SALVARE SPESSO

Quando utilizzi una versione basata su browser Spago, localmente o online, tutto il tuo sviluppo il lavoro è archiviato nella cache del browser. Questo significa che può essere perso se si cancella accidentalmente la cache, quindi è importante salvare regolarmente.

Il modo più semplice per salvare una copia di backup del tuo gioco mentre lavori su di esso è fare clic sulla freccia puntata verso l'alto nella barra in basso, a destra del nome del tuo gioco e selezionando l'opzione Publish to File dal menu a comparsa.

LE PIÙ Importanti MACRO E COMANDI

Ora che abbiamo a che fare con i passaggi di collegamento – cioè l'aspetto fondamentale di Twine - possiamo passare all'utilizzo delle macro di Harlowe e legarli insieme per aggiungere del gameplay alla nostra storia interattiva.

LE PARENTESI SONO DAVVERO IMPORTANTI

In qualsiasi linguaggio di programmazione, strutturare correttamente il codice è importante quando si tratta di rendendolo leggibile dall'uomo e garantirti di avere tutta la punteggiatura corretta.

Twine è dotato di una utilissima evidenziazione della sintassi incorporata, per rendere facile capire quale parte del tuo codice va abbinata al resto, ma tu stesso vorrai anche essere sicuro di non perdere traccia di quante parentesi quadre stai usando.

Quando si utilizzano macro condizionali come **(if :)**, **(else-if :)**, e **(else :)**, ricorda che qualsiasi cosa accada in risposta alle condizioni deve essere poste tra un paio di **[parentesi quadre]**.

È possibile nidificare più macro all'interno di esse, quindi anche se non è essenziale per il corretto funzionamento del tuo programma, è una buona idea usare tabulazioni e ritorno a capo per mantenere grandi blocchi di codice in ordine – vedi i nostri esempi per una dimostrazione.

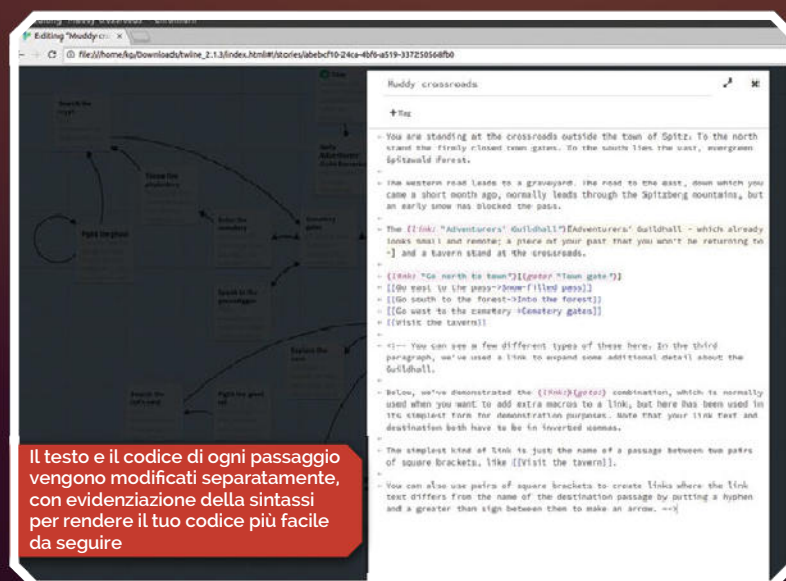
Twine salva automaticamente e istantaneamente ogni modifica, e non puoi fare affidamento sulla funzione undo (**CTRL + Z**). se accidentalmente elimini la linea che volevi tenere.

Salva regolarmente delle copie del tuo lavoro e quando lavori su una parte particolarmente complessa, è una buona idea tagliarla e incollarla in un nuovo passaggio, per rendere più semplice tornare alla versione originale, se necessario.

Il testo e il codice di ogni passaggio vengono modificati separatamente, con evidenziazione della sintassi per rendere il tuo codice più facile da seguire

LE VARIABILI (SET:)TANO LA STORIA

Nel nostro codice di creazione del personaggio (vedi "Setta le tue caratteristiche"), abbiamo usato delle macro per definire il nome del nostro eroe, i punti ferita, attacco, fortuna e qualche altra caratteristica. Iniziamo chiedendo al giocatore di dare un nome al proprio eroe, quindi useremo la macro **(set :)** per salvarlo come una variabile chiamata **$name**. La macro **(prompt :)** crea un a casella di inserimento testo. il secondo spezzone di testo tra virgolette **"Avatar"** è il testo suggerito per la casella di testo, nel caso in cui il giocatore non voglia inserire un nome.

Tutto il resto è impostato quando il il giocatore fa clic sul link 'It's time to go'. La macro **(link :)** definisce il

Setta le tue caratteristiche

```
(set: $name to (prompt: "Saluti, avventuriero. Qual'è
il tuo nome?", "Avatar"))

Quindi, $name, è la tua ultima mattinata nella caserma
della corporazione degli Avventurieri ... (testo
completo omesso per brevità).


(link: "E' ora di andare")[
    (set: $hp to 15)
    (set: $max_hp to 15)
    (set: $reputation to 0)
    (set: $atk to 11)
    (set: $luck to 7)
    (set: $battlecry to "Morirai per la mia spada!")
    (goto: "Incrocio fangoso")
```

Sfida la fortuna

```
(link: "Nuota per lui - *lancio della fortuna*")[
    (set: $roll to (random: 1, 10))
    (if: $luck > $roll )[
    Rumore di dadi. Un fortunato $roll! Nuota
fino a quando le tue braccia non sono stanche,
finché non pensi sia inevitabile che tu debba
annegare, ma in qualche modo ce la fai a uscire
alla luce del giorno e artigli la tua strada su
un sudicio banco di sabbia. (link: "Continua.")
[(goto: "Sotto ai moli")]
    ]
    (else:)[
     Rumore di dadi. Uno sfortunato $roll! Batti
il tunnel del flusso di ritorno, ma il tuo
respiro, non durerà. Ti assalgono i crampi, e
scivoli nella calda e appiccicosa presa dell'
acqua verde. (link:" Continua.") [(goto: "Sei
annegato")]
    ]
]
```

testo che vede il giocatore. Poi usiamo la macro (set:) per impostare le nostre variabili. Qui, **$luck** è impostata come numero 7. Se stai settando una variabile con una stringa di testo, come la nostra variabile **$battlecry**, devi racchiuderla tra virgolette.

L'ultima riga **(goto:)** dice al programma dove deve andare il collegamento dopo che tutto è stato impostato. D'ora in poi, ogni volta che mettiamo $name o $lucky nel testo di un passaggio, il gioco visualizzerà quelle variabili.

Quindi se creiamo un passaggio introducendo la custode della taverna, lei può dire **"Heilà, $name! Vieni su e bevi qualcosa!"**, e il nome del personaggio del giocatore apparirà nella frase.

LANCIA LE OSSA

Il lancio di dadi – o meglio, generare numeri casuali – è un fattore chiave dei giochi ed è anche molto semplice da fare usando Harlowe. Basta usare la macro (random:) con i numeri massimo e minimo dell'intervallo desiderato. Per lanciare un dado a sei facce, inserisci (random:1,6) e Twine ti darà il risultato. per renderlo parte del gioco, lo useremo contro una statistica.
Il codice "Sfida la fortuna" dimostra un modo per farlo.
Il nostro (link :) mostra al giocatore un link che dice "Nuota per lui – lancio della fortuna'. Raccontare al giocatore quando si stanno lanciando i dadi aiuta a creare la sensazione di un gamebook.

Qui, noi **(set:)**tiamo una variabile chiamata **$roll** come risultato di un numero **(random:)** compreso tra 1 e 10, per simulare un dado a dieci facce.

(if:) la fortuna **$stat** del personaggio, che abbiamo **(set:)**tato durante la creazione del personaggio, è maggiore o uguale a (> =) il numero casuale che abbiamo archiviato nella variabile **$roll** proprio ora, il nostro eroe nuota verso la salvezza, e, quando selezionano Continua, sbarca in un'area chiamata "Sotto ai moli" grazie a un comando **(goto:)**. Poiché questa è una semplice situazione “questo o quello”, possiamo usare il semplice comando **(else:)** per gestire cosa succede altrimenti. Quindi, se la fortuna del giocatore è inferiore al numero casuale, vengono inviati a un passaggio chiamato 'Sei annegato' usando un comando macro (link:) [(goto:)].

Fantasy Adventure Simulator 0.13 alpha

As the door clangs shut behind you, water starts pouring into the small room. The murky, stinking liquid is already at waist height, but you saw an an outflow pipe across the room that you might just be able to fit through.

Dice rattle. Lucky 3! You swim until your arms are tired, until you think it inevitable that you must drown, but somehow, you make it out into daylight and claw your way onto a grimy sandbank. Continue.

Fantasy Adventure Simulator 0.13 alpha

As the door clangs shut behind you, water starts pouring into the small room. The murky, stinking liquid is already at waist height, but you saw an an outflow pipe across the room that you might just be able to fit through.

Dice rattle. Unlucky 6! You strike out into the outflow tunnel but your breath won't last. Your muscles cramp, and you slip into the water's sultry green grasp. Continue.

Gli strumenti di debug integrati ti aiutano a vedere dove sta fallendo una parte del codice

FIGHT ME

Non esiste un'avventura fantasy classica senza un po' di mazzate al mostro, quindi la nostra sezione finale del codice d'esempio mostra come configurare una routine di combattimento.

La cosa bella di questo pezzo di codice è che puoi riutilizzarlo ovunque tu voglia con solo poche modifiche per cambiare il nome del mostro, impostato dalla variabile **$monster**; i suoi punti ferita, impostati da **$mob_hp**; e il suo attacco di base, definito da **$monster_atk**.

Tutte queste variabili sono state definite utilizzando il comando **(set:)** con un collegamento che va da un precedente passaggio alla sequenza di combattimento stessa. Sono bloccate quando il giocatore seleziona il link "Combatti lo scheletro".
La sequenza di combattimento è progettata come un ciclo che risponde alle decisioni del giocatore. Ogni ciclo rappresenta un round di combattimento.
Se né il personaggio del giocatore né il mostro è ridotto a zero punti ferita durante il combattimento, al giocatore verrà data l'opzione di scappare o continuare con un altro round di combattimento.

All'inizio di ogni round, il gioco tira un dado a dieci facce, sia per il giocatore che per il mostro. Il i risultati vengono aggiunti alle loro statistiche di attacco base. Chiunque abbia il punteggio più alto colpisce di due punti ferita l'altro. Se entrambi i punteggi sono uguali, niente colpi.

Nota che utilizziamo i segni maggiore di (>) o minore di (<) per la maggior parte diquesti confronti, ma se vogliamo vedere se i loro attacchi sono uguali a qualsiasi

Combatti lo scheletro

IL SETUP

Nel passaggio precedente aella sequenza di combattimento, includi questo link:

```
(link: "Fight the skeleton")[(set: $mob_hp to 10)(set:
$monster_atk to 6)(set: $monster to "The skeleton")(goto:
"Fight the skeleton")]
```

IL COMBATTIMENTO

Questa è la nostra sequenza di combattimento principale.

```
Combat against the skeleton!

(set: $hero_roll to (random: 1, 10))
(set: $mob_roll to (random: 1, 10))
(set: $hero_atk to $atk + $hero_roll)
(set: $mob_atk to $monster_atk + $mob_roll)

Dice rattle... your attack is $hero_atk. $monster's attack is
$mob_atk.

(if: $mob_atk > $hero_atk)[
    (set: $hp to $hp - 2)[$monster strikes you. $monster's HP
is $mob_hp. Your HP is now $hp.]
    (if: $hp > 0)[
        [You reel but keep your footing.]
        (link-goto: "Attempt escape", "Into the forest")
        (link-goto: "Keep fighting", "Fight the skeleton")
        ]
    (else:)[
    [YOU HAVE DIED]
    (link:"Restart")[(reload:)]
    ]
    ]

(else-if: $hero_atk > $mob_atk)[
    (set: $mob_hp to $mob_hp - 2)[Your blow hits home with a
crack of splintering bone. $monster's HP is now $mob_hp. Your
HP is $hp.]
    (if: $mob_hp > 0)[
    [$monster stumbles but regains its balance.]
        (link-goto: "Attempt escape", "Into the forest")
        (link-goto: "Keep fighting", "Fight the skeleton")
        ]
    (else:)[
    [[YOU ARE VICTORIOUS!->Search the skeleton's barrow]]
    ]
    ]

(else-if: $mob_atk is $hero_atk)[
    [You lunge at each other but both miss. $monster's HP is
$mob_hp. Your HP is $hp.]
    (link-goto: "Attempt escape", "Into the forest")
    (link-goto: "Keep fighting", "Fight the skeleton")
    ]
```

LA RICOMPENSA

Introduce un nuovo passaggio intitolato 'Search the skeleton's barrow'.

```
(set: $reputation to $reputation +1)
(set: $skeleton to "defeated")

The skeletal warrior collapses into a shower of bone shards,
just in front of a shallow barrow made of stone and earth.
Braving its musty-smelling depths, you search the tomb.

Your reputation is now $reputation.
[[Return to the forest->Into the forest]]
```

altro, dobbiamo usare la parola is, come in **(else-if: $mob_atk is $hero_atk)[outcome]**.

L'intera routine è molto semplice, ma fa il suo lavoro. Man mano che diventi più pratico di Harlowe, puoi aggiungere funzionalità extra, come ad esempio modificando la variabile **$atk**, aumentare la quantità di danni che giocatore fa se ottiene una nuova ascia scintillante, oche sia richiesto un lancio di dadi fortunato per scappare con successo dal mostro.

Se il giocatore muore, gli offriamo l'opportunità di riavviare il gioco dall'inizio. Per impostazione predefinita, Twine ha un pulsante indietro, quindi probabilmente torneranno semplicemente al punto prima che iniziasse la lotta - come la maggior parte delle persone fa con i gamebook fisici - ma il link **(reload:)** è ottimo per le persone che vogliono giocare a un gioco onesto o se usi l'editor del foglio di stile di Twine per rimuovere il tasto indietro.

Se il giocatore vince il combattimento, allora lo portiamo a una schermata di vittoria dove possiamo assegnargli il bottino, spostarlo ulteriormente lungo un percorso fisso della storia, o - se vogliamo possa essere in grado di tornare allo stesso posizione in futuro – **(set:)**tiamo una variabile per contrassegnare il nemico come sconfitto in modo che non appaia una seconda volta.

DEBUG ZAPPER

Twine ha uno strumento di debug integrato che ti aiuterà a trovare dove e perché il tuo codice stia andando male. Per eseguire il gioco prima nel debugger, usa il pulsante Test nella parte inferiore della visualizzazione della mappa principale del gioco.

L'interfaccia di test esegue il tuo gioco mentre ne traccia, in basso a destra, i turni, il titolo del passaggio e le variabili che vengono impostate. Per vedere maggiori dettagli del codice, premi il pulsante Debug View. Puoi anche testare i singoli passaggi tramite l'icona bug che appare quando si muove il mouse sopra qualsiasi passaggio, ma ricordati che funzionerà senza qualsiasi variabile che potresti avere settato nei passaggi precedenti.

AGGIUNGI MULTIMEDIA

PUBLBLICA IL TUO GIOCO

TUTTI CANTANO, TUTTI BALLANO

Twine 2 ha la reputazione di essere meno multimedia-friendly di Twine 1, che potrebbe codificare le immagini come parte del file di gioco, ma è davvero molto semplice usare l'HTML per includere immagini, suoni e persino video.

Per aggiungere delle immagini e unirsi allo sviluppo di lunga tradizione delle avventure testo e grafica, semplicemente usa il tag HTML <img> per visualizzare immagini e il tag <audio> per il suono.

Se vuoi pubblicare il tuo gioco online, il suono e le immagini dovranno devono essere ospitati da qualche parte. In questo tutorial, useremo **itch.io** per pubblicare il nostro gioco, così può ospitare, per te, le tue immagini. Significa che puoi semplicemente inserirle nella stessa directory del tuo file di gioco e usare il nome del loro file nel tuo codice per riferirsi a loro, piuttosto che un URL completo.

Puoi anche incorporare video YouTube, che può essere efficace se vuoi aggiungere filmati di intermezzo a una avventura fantasy o interviste con i sospetti in un poliziesco. Carica il tuo video su YouTube, copia il codice embed dalle Opzioni di condivisione di YouTube e incollalo nel passaggio Twine in cui vuoi che appaia.

PUBBLICA E SII FELICE, IN REALTA'

La bellezza di un gioco di Twine è che esiste come un singolo file HTML che puoi caricare ovunque tu abbia un po' di spazio web. Esistono anche host gratuiti specificamente dedicati agli sviluppatori con Twine.

Tuttavia, se vuoi un set di strumenti professionale per pubblicare e persino monetizzare i tuoi giochi, ti consigliamo di registrare un account gratuito su **itch.io**.

Anche se vuoi distribuire i tuoi giochi gratuitamente, configura la casella di donazione per i tuoi fan, o puoi rilasciare i giochi commercialmente, gli strumenti di **itch.io** rendono il lavoro semplice.

Incorpora multimedia con HTML

Il codice HTML può essere inserito ovunque all'interno di un passaggio.

Incorpora e centra un'immagine:

```
<center><img src="yourimage.jpg"; alt="victory gate"></center>
```

Incorpora un file audio:

```
<audio src="youraudiofile.mp3"; autoplay>
```

Incorpora un video di YouTube:

```
<iframe width="560" height="315"
src="yourYouTubevideoURL" frameborder="0"
allowfullscreen></iframe>
```

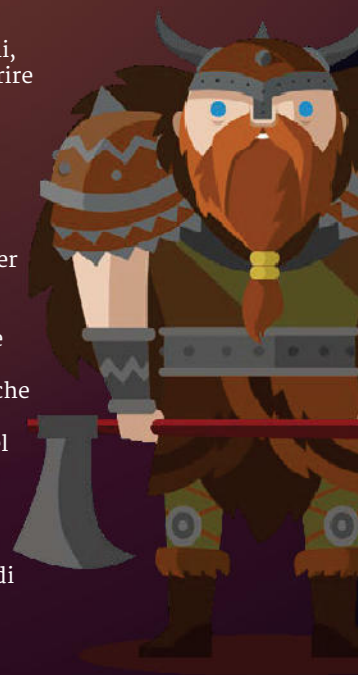
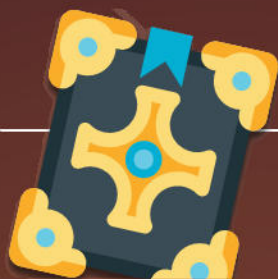
Oppure utilizza le opzioni di condivisione e incorporamento di YouTube per generare automaticamente il codice corretto.

Itch.io fornisce strumenti gratuiti per pubblicare, vendere e persino monetizzare i tuoi giochi

Puoi caricare blog, screenshot e video per mantenere i giocatori aggiornati, è senza pubblicità, e può persino gestire per te i pagamenti con le tasse (VAT MOSS) sui tuoi profitti.

Una volta che hai creato un account, vai su My Dashboard e premi 'Create new project'. Da li, tutto quello che devi fare è inserire un titolo, una descrizione, un'immagine di copertina e caricare il file HTML. Ricorda di spuntare il 'This file will be played in the browser', quindi abilita lo schermo intero 'Mobile friendly', e le opzioni per le barre di scorrimento, più in basso nella pagina.

Se hai della grafica o audio che vuoi includere, mettili in una directory con il tuo file HTML, che deve chiamarsi **index.html**, comprimi tutto e fai l'upload del file zip.

Puoi vedere l'anteprima della pagina del gioco come versione bozza per essere certo tutto funzioni correttamente, prima di premere Publish per pubblicare il tuo capolavoro.

PROGRAMMA UNA AVVENTURA TESTUALE in PYTHON

Programmare una avventura testuale è un classico modo di imparare. Scopri come creare la tua avventura con Python

Cosa Serve

- Raspberry Pi
- Python
- Delle idee

Ora che hai imparato a progettare e a costruire un'avventura testuale, è ora di imparare a programmarti la tua. In questo progetto, progetterai e programmerai il tuo gioco RPG labirinto. Lo scopo del gioco sarà quello di raccogliere oggetti e fuggire da una casa, assicurandosi di evitare tutti i mostri.

Questo progetto insegna il game design attraverso lo sviluppo di un gioco RPG labirinto. In questo gioco, il giocatore deve raccogliere oggetti all'interno di una casa e raggiungere una stanza specifica, evitando i mostri in agguato in alcune stanze. Questo gioco sarà ottenuto manipolando dizionari e liste.

Inizia aprendo Thonny (Programmazione> Thonny Python IDE) e inserisci il listato **rpg.py**. Questo è un gioco RPG molto semplice che ha solo due stanze. In **Figura 1** una mappa del gioco base.

Scegli File> Save e salva il codice come **rpg.py**. Fai clic sull'icona Run per testare il codice. Puoi digitare **go south** per spostarti dalla sala alla cucina, e poi **go north** per tornare di nuovo nella hall! Digita go west nella hall e otterrai un amichevole messaggio d'errore: "Non puoi andare da quella parte!"

Fai clic su Stop e guarda questa parte del codice:

```
#a dictionary linking a room to other rooms
rooms = {

'Hall' : { 'south' : 'Kitchen'
          },

'Kitchen' : { 'north' : 'Hall'
            }
        }
```

Ogni stanza è un dizionario e le stanze sono collegate insieme usando le direzioni. Aggiungiamo una sala da pranzo alla tua mappa, ad est della sala (come mostrato nella **Figura 2**).

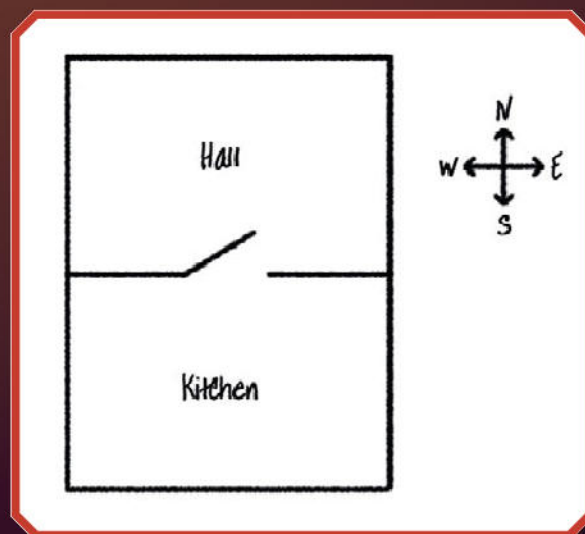



Figura 1 Mappa del gioco di base

rpg.py

magpi.cc/2jbnPol

```
01. #!/bin/python3
02.
03. def showInstructions():
04.   #stampa il menu principale e i comandi
05.   print('''
06. RPG Game
07. ========
08. Comandi:
09.   go [direzione]
10.   get [oggetto]
11. ''')
12.
13. def showStatus():
14.   #stampa lo stato corrente del giocatore
15.   print('---------------------------')
16.   print('Sei in ' + currentRoom)
17.   #stampa l'inventario corrente
18.   print('Inventario : ' + str(inventory))
19.   #stampa un oggetto quando presente
20.   if "item" in rooms[currentRoom]:
21.     print('Vedi un ' + rooms[currentRoom]['item']) )
22.   print("---------------------------")
23.
24. #Un inventario, inizialmente vuotio
25. inventory = []
26.
27. #Un dizionario che collega le stanze una all'altra
28. rooms = {
29.
```

Figura 2 Adesso abbiamo aggiunto una terza stanza al gioco

Devi aggiungere una terza stanza, Dining Room (Sala da Pranzo). Devi anche collegarlo alla sala (Hall) ad ovest. Inoltre, è necessario aggiungere dati alla sala, in modo che tu possa spostarti nella sala da pranzo andando a est.

```
#Un dizionario che collega le posizioni delle
stanze tra di loro
rooms = {

'Hall' : { 'south' : 'Kitchen',
           'east'  : 'Dining Room',
                },

'Kitchen' : { 'north' : 'Hall',
                },

'Dining Room' : { 'west'  : 'Hall',
             }

        }
```

Prova il gioco con la tua nuova sala da pranzo. Ora puoi andare ad est e andare ad ovest tra la sala e la sala da pranzo. Se non riesci ad entrare ed uscire dalla sala da pranzo, controlla di aver aggiunto tutto il codice sopra (inclusa la virgola extra dopo la sezione Kitchen del codice).

AGGIUNGERE OGGETTI

Aggiungere un oggetto in una stanza è semplice: basta aggiungerlo al dizionario della stanza. Mettiamo ad esempio una chiave nella hall.

```
#Un dizionario che collega le posizioni delle
stanze tra di loro
rooms = {

'Hall' : { 'south' : 'Kitchen',
           'east'  : 'Dining Room',
           'item'  : 'key'
                },
```

```
30.             'Hall' : {
31.                   'south' : 'Kitchen'
32.                 },
33.
34.             'Kitchen' : {
35.                   'north' : 'Hall'
36.                 }
37.
38.          }
39.
40. #Fai partire il giocatore dalla Hall
41. currentRoom = 'Hall'
42.
43. showInstructions()
44.
45. #ciclo infinito
46. while True:
47.
48.   showStatus()
49.
50.   #ottieni la prossima "mossa" del giocatore
51.   #.split() lo suddivide in una lista di array
52.   #ad esempio scrivendo 'vai est' darebbe la lista:
53.   #['vai','est']
54.   move = ''
55.   while move == '':
56.     move = input('>')
57.
58.   move = move.lower().split()
59.
60.   #se scrivono "vai" prima
61.   if move[0] == 'vai':
62.     #controlla che sia possibile andare dove vogliono
63.     if move[1] in rooms[currentRoom]:
64.       #imposta la stanza corrente nella nuova stanza
65.       currentRoom = rooms[currentRoom][move[1]]
66.     #non c'è una porta (collegamento) alla nuova stanza
67.     else:
68.         print('Non puoi andare in questa direzione!')
69.
70.   #se digitano 'prendi' prima
71.   if move[0] == 'prendi' :
      #se la stanza contiene un oggetto, e l'oggetto è
72.     #quello che vogliono prendere
      if "item" in rooms[currentRoom] and move[1] in
73. rooms[currentRoom]['item']:
74.       #aggiungi l'oggetto all'inventario
75.       inventory += [move[1]]
76.       #mostra un utile messaggio
77.       print(move[1] + ' preso!')
78.       #cancella l'oggetto dalla stanza
79.       del rooms[currentRoom]['item']
80.     #altrimenti, se l'oggetto da prendere non c'è
81.     else:
82.       #digli che non puoi prenderlo
83.       print('Non posso prenderlo ' + move[1] + '!')
```

rpg-finished.py

magpi.cc/2jbnPol

```
001. #!/bin/python3
002.
003. def showInstructions():
004.     #stampa il menu principale e i comandi
005.     print('''
006. RPG Game
007. ========
008.
009. Raggiungi il giardino con una chiave e una pozione
010. Evita i mostri!
011.
012. Comandi:
013.   vai [direzione]
014.   prendi [oggetto]
015. ''')
016.
017. def showStatus():
018.   #stampa lo stato corrente del giocatore
019.   print('---------------------------')
020.   print('Sei in ' + currentRoom)
021.   #stampa l'inventario corrente
022.   print("Inventario : " + str(inventory))
023.   #stampa un oggetto se presente
024.   if "item" in rooms[currentRoom]:
025.     print('Vedi un ' + rooms[currentRoom]['item'])
026.   print("---------------------------")
027.
028. #Un inventario, inizialmente vuoto
029. inventory = []
030.
031. #Un dizionario che collega tra loro le stanze
032. rooms = {
033.
034.             'Hall' : { 'south' : 'Kitchen',
035.                   'east'  : 'Dining Room',
036.                   'item'  : 'key'
037.                 },
038.
039.             'Kitchen' : { 'north' : 'Hall',
040.                   'item'  : 'monster'
041.                 },
042.
043.             'Dining Room' : { 'west'  : 'Hall',
044.                   'south' : 'Garden',
045.                   'item'  : 'potion'
046.
047.                 },
048.
049.             'Garden' : { 'north' : 'Dining Room' }
050.
051.           }
052.
053. #Comincia col giocatore nella Hall
054. currentRoom = 'Hall'
055.
056. showInstructions()
057.
058. #ciclo infinito
059. while True:
```

Ricordati di mettere una virgola dopo la riga sopra il nuovo oggetto (qui dopo "Dinning Room") o il tuo programma, non funzionerà.

Se esegui il gioco dopo aver aggiunto il codice indicato prima, puoi vedere una chiave nella sala e puoi persino raccoglierla (digitando get key), e aggiungerla al tuo inventario.

AGGIUNGERE NEMICI

Questo gioco è troppo facile! Aggiungiamo dei nemici, ad alcune stanze, che il giocatore deve evitare.

Aggiungere un nemico a una stanza è facile come aggiungere qualsiasi altro oggetto. Aggiungiamo un mostro affamato alla cucina:

```
#Un dizionario che collega tra loro le posizioni delle stanze
rooms = {

'Hall' : { 'south' : 'Kitchen',
           'east'  : 'Dining Room',
           'item'  : 'key'
         },

'Kitchen' : { 'north' : 'Hall',
              'item'  : 'monster'
            },

'Dining Room' : { 'west'  : 'Hall',
                  'item'  : 'potion'
                }
```

Vuoi assicurarti che il gioco finisca se il giocatore entra in una stanza con un mostro. Puoi farlo con il seguente codice, che dovresti aggiungere alla fine del gioco:

```
  # il giocatore perde se entra in una stanza con un mostro
  if 'item' in rooms[currentRoom] and 'monster' in rooms[currentRoom]['item']:
    print('Un mostro ti ha catturato... GAME OVER!')
    break
```

Questo codice controlla se c'è un oggetto nella stanza – e se è così, se quell'oggetto è un mostro. Si noti che questo codice è rientrato, mettendolo in linea con il codice sopra. Ciò significa che il gioco lo farà controllerà la presenza di un mostro ogni volta che il giocatore entra una nuova stanza.

Prova il tuo codice andando in cucina, che ora contiene un mostro. Digita **go south** e vedrai la scritta "Un mostro ti ha catturato... GAME OVER!"

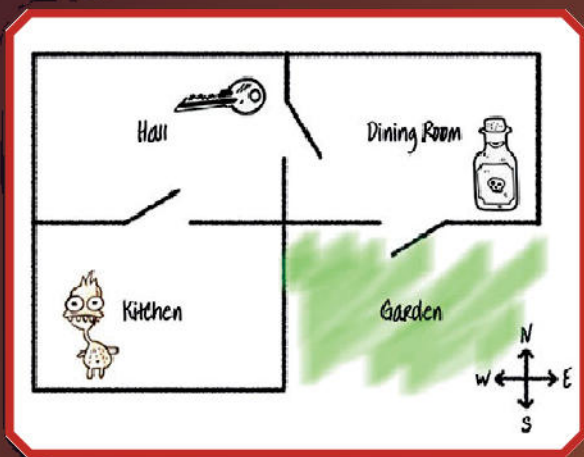



Figura 3 La mappa finale del gioco d'avventurta

VINCERE IL GIOCO

Diamo al tuo giocatore una missione che deve essere completata per vincere il gioco.

In questo gioco, il giocatore vince raggiungendo il giardino e scappando dalla casa. Deve anche avere la chiave e la pozione magica con sé. La Figura 3 mostra una mappa finale del gioco.

Per cominciare, devi aggiungere un giardino a sud della sala da pranzo. Ricorda di aggiungere le porte, per collegare le altre stanze della casa.

Aggiungi una pozione alla sala da pranzo (o a un'altra stanza nella tua casa). Infine, aggiungi questo codice per consentire al giocatore di vincere la partita quando arriva in giardino con la chiave e la pozione:

```
  # il giocatore vince se arriva in giardino
con una chiave e uno scudo
  if currentRoom == 'Garden' and 'key' in
inventory and 'potion' in inventory:
    print('Sei uscito dalla casa.. HAI VINTO!')
    break
```

Puoi vedere il codice finale in **rpg-finished.py**.

Infine, verifica il tuo gioco per assicurarti che il giocatore può vincere! Inserisci i seguenti comandi:

```
get key (prendi chiave)
go east (vai a est)
get potion (prendi la pozione)
go south (vai a sud)
```

Vedrai 'sei uscito dalla casa... HAI VINTO!'

Usa quello che hai imparato per creare il tuo gioco. Aggiungi un sacco di stanze, mostri da evitare e oggetti da raccogliere. Ricordati di modificare il codice in modo che il giocatore vince quando arriva ad una certa stanza con alcuni degli oggetti nel proprio inventario. Potrebbe aiutarti traccia una mappa, prima di iniziare a scrivere il codice.

```
060.
061.   showStatus()
062.
063.   #ottieni la prossima "mossa" del giocatore
064.   #.split() lo suddivide in una lista di array
065.   #ad esempio scrivendo 'vai est' darebbe la lista:
066.   #['vai','est']
067.   move = ''
068.   while move == '':
069.     move = input('>')
070.
071.   move = move.lower().split()
072.
073.   #se scrivono "vai" prima
074.   if move[0] == 'vai':
075.     #controlla che sia possibile andare dove vogliono
076.     if move[1] in rooms[currentRoom]:
077.       #imposta la stanza corrente nella nuova stanza
078.       currentRoom = rooms[currentRoom][move[1]]
079.     #non c'è una porta (collegamento) alla nuova stanza
080.     else:
081.       print('Non puoi andare in questa direzione!!')
082.
083.   #se digitano 'prendi' prima
084.   if move[0] == 'prendi' :
085.     #se la stanza contiene un oggetto, e l'oggetto è
       #quello che vogliono prendere
086.     if 'item' in rooms[currentRoom] and move[1] in
rooms[currentRoom]['item']:
087.       #aggiungi l'oggetto all'inventario
088.       inventory += [move[1]]
089.       #mostra un utile messaggio
090.       print(move[1] + ' preso!')
091.       #delete the item from the room
092.       del rooms[currentRoom]['item']
093.     #altrimenti, se l'oggetto da prendere non c'è
094.     else:
095.       #digli che non puoi prenderlo
096.       print('Non posso prenderlo ' + move[1] + '!')
097.
098.   # il giocatore perde se entra in una stanza con un mostro
099.   if 'item' in rooms[currentRoom] and 'monster' in
rooms[currentRoom]['item']:
100.     print('Un mostro ti ha catturato... GAME OVER!')
101.     break
102.
103.   # il giocatore vince se arriva in giardino con una chiave
e uno scudo
104.   if currentRoom == 'Garden' and 'key' in inventory and
'potion' in inventory:
105.     print('Sei uscito dalla casa.. HAI VINTO!')
106.     break
```